Improving Reinforcement Learning Speed for Robot Control.

Laetitia Matignon, Guillaume J. Laurent, Nadine Le Fort-Piat

Improving Reinforcement Learning Speed for Robot Control

Laetitia Matignon, Guillaume J. Laurent, Nadine Le Fort-Piat
Laboratoire d'Automatique de Besançon UMR CNRS 6596
24 rue Alain Savary, 25000 Besançon, France
E-mails : (laetitia.matignon,guillaume.laurent,nadine.piat)@ens2m.fr
Web site : lab.cnrs.fr

***Abstract*—Reinforcement Learning (RL) is an intuitive way of programming well-suited for use on autonomous robots because it does not need to specify how the task has to be achieved. However, RL remains difficult to implement in realistic domains because of its slowness in convergence. In this paper, we develop a theoretical study of the influence of some RL parameters over the learning speed. We also provide experimental justifications for choosing the reward function and initial Q-values in order to improve RL speed within the context of a goal-directed robot task.**

***Index Terms*—goal-directed reinforcement learning, reward function, initial Q-values, goal bias, progress estimators**

I. Introduction

Reinforcement learning (RL) [1] is a class of *learning from experience* suitable for robotics when *on line learning without information about the environment* is required. For each of its state, the controller can learn which of its available actions will result in the best performance for a given task. For example, if the mobile robot collides with an obstacle, it will learn that it is a bad action, but if it reaches the goal, it will learn it is a good action. Such feedback from the environment is named *reinforcement* or *reward*. The controller's aim is to maximise its expected future rewards for state-action pairs, represented by the action values. Q-learning [2] is a commonly form of RL where the optimal policy is learned implicitly in the form of a Q-function.

Numerous applications of RL on robotic systems have been published. Among successful applications of RL are the learning of the motion of standing up from a chair by humanoid robots [3] or the control of a stable altitude loop of an autonomous quadrotor [4]. RL was also used to control a micro-manipulator system [5].

However, RL has an inherent problem : its *learning time increases exponentially with the size of the state space*. Consequently, RL remains difficult to implement in realistic domains, typically the robotics domain. Thus, several methods have been proposed to speed up RL through *the incorporation of prior knowledge or bias* into RL.

One example [6] is to add prior knowledge by a human operator who supplies sample robot trajectories. During this passive phase, the robot is simply using examples to boot the value function. Guidance can also be applied by an experienced robot whose Q-values are accessible by the learning robot (imitative reinforcement [7]).

Another way is to modify or *shape* the reward signal. Mataric [8] proposed a methodology for designing reward functions that take advantage of implicit domain knowledge. This methodology involves the use of continuous reward functions and *progress estimators*. Likewise, with *reward shaping*, the rewards from the environment are augmented with additional rewards. An application is the problem of learning to balance on a bicycle [9]. However, reward shaping can lead the controller into learning sub-optimal policies and so, traps the system.

Wiewora [10] completed the reward shaping study and moreover, proved certain similarities between potential-based shaping and initial Q-values. Indeed, the most elementary method for biasing learning is to choose the initial Q-values. Brief studies of the effect of this idea can be found in [11].

In the above examples, methods consist in *embedding the bias* in the reward function or the initial action value function. These RL parameters play an important part in RL. Nevertheless, although RL has been studied extensively and its convergence properties are well known, in practice, people often choose reward function on one's intuition and initial Q-values arbitrarily [1].

We propose to define some rules to choose these both RL parameters in a cautiously way. Indeed, initial Q-values do not depend only on an individual decision, but are influenced by some properties. In this paper, we discuss the effects of RL parameters on the policy in order to suggest a generic analysis. We validate our analysis with Q-learning algorithm. The main issue is to shed light on *how to correctly initialize RL parameters in order to obtain the desired optimal behavior in a minimal time within the context of a goal-directed robot task*, in order to efficiently implement RL on real robots.

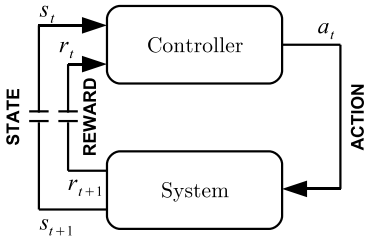

Fig. 1. Sensory motor loop.

II. Reinforcement Learning

RL is a machine learning paradigm which does not require a model of the dynamics of the environment. It is based on the *trials and errors principle*. At each time step, the controller chooses an action a_t that leads the system from the state s_t to the new state s_{t+1} [1]. Then it receives a reward r_{t+1} according to s_t, a_t and s_{t+1} (Fig. 1). The goal of the controller is to learn a mapping from states to actions called a *policy* which maximises the expected sum of discounted rewards received over time [2]. The RL algorithm that we use in this work is Q-learning [2]. The Q-learning optimal action-value function is :

$$Q^*(s,a) = E\left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right] \quad (1)$$

This represents the expected value of the reward for taking action a from state s, ending up in state s', and then acting optimally. $\gamma \in [0;1[$ is the discount factor. Q-values are typically stored in a tabular representation. Q-learning is an *off-policy* method, *i.e.* the optimal action-value function Q^* is directly approximated, independently of the policy being followed. Its updating rule is :

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \quad (2)$$

where r is the reward received for the transition from the state s to the new state s' by executing the action a. $\alpha \in]0;1]$ is the learning-rate parameter. Under some conditions [12], Q-learning algorithm is guaranteed to converge to the optimal value function Q^*. The Q-Learning algorithm chooses the action according to an exploration / exploitation criteria. We used the *ϵ-greedy method*, in which the probability of taking a random action is ϵ and, otherwise, the selected action is the one with the largest Q-value in the current state [3].

III. Choice of uniform initial Q-values with a binary reward function

We make the assumption that two general trends stand out : *the global policy* and *a specific behavior at the beginning* of the learning process. In this section, we are going to set out that these both tendencies depend on the shape of the reward function and on the initialization of the action-value function. We first considered *a binary reward function* which has the advantage to include a lot of cases and it is possible to extrapolate.

A. Optimal policy

The binary reward function is such as the reward received is always r_∞ except if the new state is the goal state and then, the reward is r_g. It's given by :

$$\forall s \in S\ \forall a \in A,\ R(s,a,s') = \begin{cases} r_g \text{ if } s' = s_g \\ r_\infty \text{ else} \end{cases} \quad (3)$$

where s' is the state obtained by executing the action a from s, and s_g the goal state.
In case of all rewards are identical ($r_g = r_\infty$), the solution of the equation (1) is a constant noted Q_∞,

$$\forall s\ \forall a\ Q^*(s,a) = Q_\infty = \frac{r_\infty}{1-\gamma} \quad (4)$$

That is to say that during the learning process, Q-values for all state-action values converge to Q_∞.

Nevertheless, if $r_g \neq r_\infty$, Q_∞ is the limit of the action-value function $Q^*(s,a)$ when the distance between s and s_g tends toward infinity. So, toward the goal, states are *more and more or less and less attractive* depending on r_g and Q_∞. On the one hand, if $r_g > Q_\infty$, the Q-value of state-action pairs moving to the goal state will be more and more attractive than Q_∞. So the global optimal policy is *the shortest way* toward the goal state. On the other hand, if $r_g < Q_\infty$, the optimal policy is random everywhere except *a local repulsion of s_g* near the goal state.

Of course, the shortest way toward the goal is the sought optimal policy within our context of reaching a goal in a minimal time. *So r_g must always be superior to Q_∞.*

B. Behavior at the beginning of the learning process

As well as the reward function, the initial value Q_i of the action-value function has an effect on the policy, but only at the beginning of the learning process. We believe that a global trend can be underscored during the first trials of the learning process.

Let's examine what the Q-values are expected to be at the beginning. If we calculate the first updating of a state-action pair (s,a) thanks to the equation (2), such as the next state s' is not the goal state and has not been updated, we have :

$$\begin{aligned} Q(s,a) &\leftarrow Q_i + \alpha\left[r_\infty + (\gamma - 1)Q_i\right] \\ &\leftarrow Q_i + \alpha(1-\gamma)(Q_\infty - Q_i) \end{aligned} \quad (5)$$

So the discriminating value of Q_i is also Q_∞. According to the value of Q_i compared to Q_∞, *states already visited will be more or less attractive* as long as the controller has not reached plenty of times the goal state.

[1] The framework is defined as a finite set of states S and a finite set of actions A.

[2] The rewards are discounted by a discount factor γ that controls the balance between the significance of immediate rewards and future rewards.

[3] If several values are identical, the choice will be random among greedy actions.

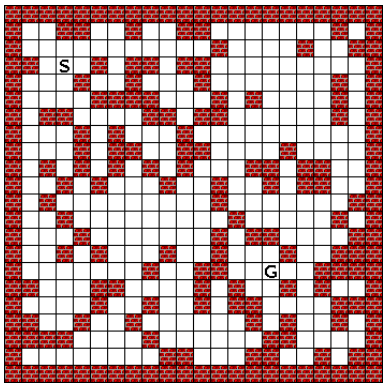

Fig. 2. Non-deterministic 20×20 maze with a single start state in the upper left corner (S) and a goal state in the opposite corner (G).

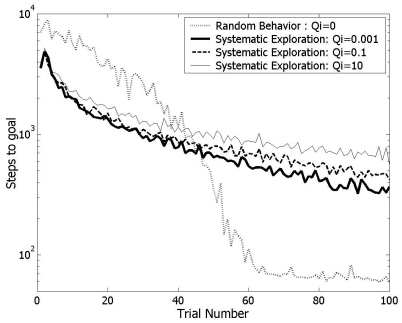

Fig. 3. Maze experiments with different Q_i averaged over 50 independent runs. Steps to goal *vs.* trial number. It illustrates random and systematic exploration as behavior at the beginning of the process.

- **If** $Q_i > Q_\infty$ **:** states which have already been visited will have a value lower than the value of states which haven't yet been visited ($Q(s,a) < Q_i$). In other words, states which haven't yet been visited will be more attractive. It induces the controller *to explore* more systematically at the beginning of the learning than a random exploration. We called it *systematic exploration behavior*.
- **If** $Q_i < Q_\infty$ **:** states which have already been visited will have a value superior to the value of states which haven't yet been visited ($Q(s,a) > Q_i$). *i.e.* states which have already been visited will be more attractive. It leads the controller into less exploration at the beginning, which considerably slows down the learning. We named this behavior "*moving round in circles*". Of course, it's better to avoid it.
- **If** $Q_i = Q_\infty$ **:** states which have already been visited will have the same value than states which haven't yet been visited ($Q(s,a) = Q_i$). So we obtain at the beginning *a pure random behavior*.

C. Maze experiments

We have discussed how traditional reward functions and arbitrary initial Q-values may slow down the learning of an interesting policy. At this point, we are going *to validate this previous analysis* and we have chosen for simplicity and clarity to use at first a non-deterministic maze domain to demonstrate how the robot's behavior is influenced by using binary rewards and different initial Q-values.

1) Task: The system is represented by a mobile robot which has to travel through a maze-like environment (Fig. 2). Each robot's position is a discrete state. When the robot reaches the goal state, the trial ends. The robot chooses from four actions, representing an intention to move in one of the four cardinal directions (N,E,S,W). An action that would move the robot in a wall instead leaves the robot in its current position. Any movement moves the robot in the intended direction with probability 0.6, and otherwise in a random state of the four neighboring states of the expected state. All trials use Q-learning with a learning-rate $\alpha = 0.1$, a discount factor $\gamma = 0.9$, a tabular Q-table initialized uniformly Q_i and follow the ϵ-greedy method ($\epsilon = 0.1$).

2) Binary reward function: Our reward function replicates the function given in (3), with $r_g = 1$ and $r_\infty = 0$. With such a reward function, the optimal policy is *the shortest way toward the goal* and the discriminating value of Q_i is 0 (Q_∞=0). Fig. 3 illustrates our previous analysis. As can be readily seen, using *the systematic exploration behavior* helped speed up learning during the first trials. Indeed, the robot visited every nook and cranny of the complex maze and the goal state s_g was discovered faster than in case of a random exploration. On the other hand, in case of systematic exploration, the robot was always spurred on to explore and that's why it always took more steps to reach the goal after some trials. Besides, we used different values of Q_i for the systematic exploration and we notice the more Q_i was superior to Q_∞, the more the robot explored. So, *the more the difference between Q_i and Q_∞ is important, the more the general behavior is underlined.*

Concerning the *moving round in circles behavior* ($Q_i < 0$), we do not submit any experiments. Indeed, the robot took too much time to reach the goal given that it moved round in circles in an area where Q updates are in the shape of γ^n, with n the trial number. So the action-value function in the moving round area tends towards zero, which is a well approximated value [4]. Anyway, *this behavior has to be avoided.*

D. Conclusion

These experiments shed light on *the importance of initial Q-values*. The choice of Q_i is not trivial and must be carried out according to the desired behavior. When the system naturally goes away from the goal, a systematic exploration should be preferred so as to speed up learning at the beginning. Indeed, systematic exploration forces the controller to explore, and so to approach the goal.

[4] the break point is around $-1.7e^{-308}$.

IV. Choice of continuous reward function and heterogeneous initial Q-values

In order to broaden the scope of our study, we propose henceforth to enlarge the study by using first two different *continuous reward functions* with uniform initial Q-values, and secondly by initializing the action value function with *a goal bias function*.

A. Reward function using progress estimators

First, we propose to study the use of a continuous reward function instead of binary rewards. In this view and to speed up the learning, some authors introduce reward functions by using *progress estimators* [8] or potential-based shaping [10]. Progress estimators provide a measure of improvement relative to an objective. They do not supply a complete information but only partial, goal-specific "advice". The aim of the learning process is to maximize the progress function $\varphi : S \times A \rightarrow \mathbb{R}$. For instance, concerning the maze, the progress estimator can be an assessment of the expected number of steps needed to get to the goal from the new state s', defined as $\varphi(s', a) = d(s', s_g)$. d is the manhattan distance between the new state s' and s_g. The aim of the controller in the maze is to minimize this function and the parameters could be then :

$$\begin{cases} \mathrm{r}(s, a, s') = -\varphi^2 = -d^2(s', s_g) \\ Q_i = 0 \end{cases} \tag{6}$$

Thus, the controller is less and less punished by approaching the goal. The global policy is the shortest way toward the goal. Given our maze, this reinforcement is spurious as there are plenty of walls between the initial state and the goal. In particular, it entails an *unlearning phenomenon* after few trials. Indeed, if the robot was taken off toward a dead end (that moves the robot closer to the goal), it would get out of the trap only thanks to exploration because states are more and more attractive toward the goal. At the beginning, Q-values are near 0 so systematic exploration is strong : going out of the trap is possible. But after few trials, turning back is tantamount to choosing a less attractive Q-value and will happen only if several exploration actions follow one another, *i.e.* seldom.

So, both progress estimators and potential-based shaping are risky. It's better to use cautiously these approaches insofar as they may lead to a pernicious behavior that catches the system out.

B. Continuous reward function inspired by gaussian function

Consequently, we propose a continuous reward function such that on the one hand, r *is uniform* for some states far from the goal in order to avoid the unlearning phenomenon, and on the other hand, there is *a reward gradient* in a zone around the goal. We suggest the reward function inspired by *the gaussian function* :

$$\mathrm{r}(s, a, s') = \beta e^{-\frac{d(s', s_g)^2}{2\sigma^2}} \tag{7}$$

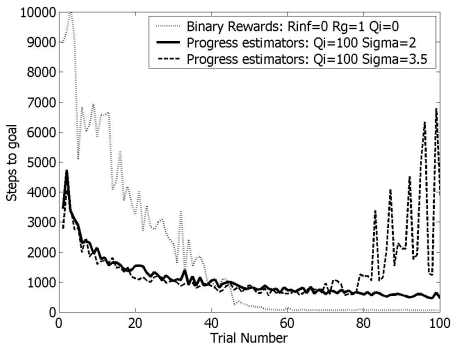

Fig. 4. Maze experiments with reward function inspired by gaussian function averaged over 20 independent runs. Steps to goal *vs.* trial number. $Q_i = 100$; $\mathrm{r}(s, a, s') = 10e^{-\frac{d(s', s_g)^2}{2\sigma^2}}$.

Q_i values are uniform, β adjusts the amplitude of the function and σ, the standard deviation, specifies the *reward gradient influence area*. As a matter of course, "moving round in circles" behavior must be avoided by choosing $Q_i \geq \frac{\beta}{1-\gamma}$.

For the maze task, we have chosen the continuous reward function $Q_i = 100$ and $\beta = 10$. Fig. 4 shows the unlearning phenomenon as from 80 trials with $\sigma = 3.5$ [5]. Indeed, with such σ, the reward gradient influence area is too large and includes few dead ends. On the contrary, if the reward gradient influence area is only 6 steps around the goal ($\sigma = 2$), there won't be any unlearning phenomenons and the learning process is accelerated.

Such a continuous reward function is *adjustable* in order to avoid a harmful behavior. Anyway, the best approach would be to influence fleetingly the learning process.

C. Goal Bias

In view of the importance of the action-value function initialization, we propose as a matter of course to be inspired by *progress estimators* in order to *initialize the action-value function* with more precise information. In this section, the reward function is the binary one given by (3) with $r_\infty = 0$ and $r_g = 1$.

We are going to settle a correct goal bias function thanks to our previous analysis. An interesting bias shall achieve *an adjustable state gradient* and in addition, must avoid the "moving round in circles" behavior. We suggest for instance *a gaussian goal bias function* :

$$Q_i(s, a) = \beta e^{-\frac{d(s, s_g)^2}{2\sigma^2}} + \delta + Q_\infty. \tag{8}$$

δ fixes the level of systematic exploration far from the goal, β the amplitude of the bias and σ the bias influence area.

Concerning the maze, the bias is such that states near the goal are more and more interesting *a priori* than states far away from the goal. So δ and β must be chosen very small compared to one (in order to avoid too much

[5] *i.e.* states 10 steps away from the goal have uniform r.

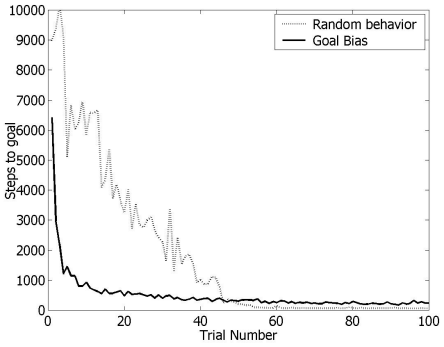

Fig. 5. Maze experiments with goal bias function averaged over 20 independent runs . Steps to goal *vs.* trial number with binary rewards. Random test is $Q_i = 0$. Goal bias function is $Q_i(s,a) = 0.001e^{-\frac{d(s,s_g)^2}{2\times 13^2}}$.

systematic exploration). Fig. 5 presents goal bias results on the previous maze which are unambiguous. *The goal bias leads to a much faster learning process.* It is worth noticing that there is *no problem concerning dead ends even if the bias is wrong.* Contrary to Sect. IV-B, the effect of the goal bias function is transient. It advises the controller *only at the beginning of the learning process.*

D. Conclusion

Both potential-based shaping and progress estimators methods must be used cautiously to design a continuous reward function. Consequently, we have proposed a continuous reward function inspired by a gaussian function and whose reward gradient influence area is adjustable in order to deal with risks. *Anyway, the best solution is to choose a suitable goal bias function that does not lead to any problems. Our analysis helps the choice of a correct goal bias function.*

V. Experiments with the pendulum swing-up task

Last of all, we validate our results on the continuous space control task of *a pendulum swinging upwards with limited torque* [13] (Fig. 6). The control of this one degree of freedom system is non-trivial if the maximal output

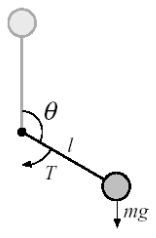

Fig. 6. A pendulum with limited torque. The dynamics were given by $\dot{\theta} = \omega$ and $ml^2\dot{\omega} = -\mu\omega + mgl\sin\theta + u$. The physical parameters were $m = l = 1$, $g = 9.8$, $\mu = 0.01$, and $u^{max} = 5.0$. The learning parameters were $\gamma = 0.97$, $\alpha = 0.1$.

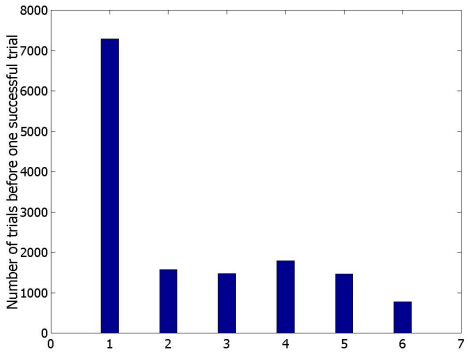

Fig. 7. Comparison of number of trials before one successful trial. The simulation lasted 10000 trials averaged over 10 independent runs. `bar1:` {binary reward function ; $Q_i(\mathbf{x}) = 0$}
`bar2:` {binary reward function ; $Q_i(\mathbf{x}) = 0.1$}
`bar3:` {binary reward function ; $Q_i(\mathbf{x}) = e^{-\frac{\theta^2}{2\times 0.25^2}} + 0.1$ }
`bar4:` $\{R(\mathbf{x},u,\mathbf{x}') = \cos(\theta')\ ;\ Q_i(\mathbf{x}) = 0\}$
`bar5:` $\{R(\mathbf{x},u,\mathbf{x}') = e^{-\frac{\theta'^2}{2\times 0.25^2}}\ ;\ Q_i(\mathbf{x}) = 10\ \}$
`bar6:` $\{R(\mathbf{x},u,\mathbf{x}') = e^{-\frac{\theta'^2}{2\times 0.25^2}}\ ;\ Q_i(\mathbf{x}) = 11\ e^{-\frac{\theta^2}{2\times 0.25^2}} + 0.1\}$

torque u^{max} is smaller than the maximal load torque mgl. The controller has to swing the pendulum several times to build up enough momentum to bring it upright and has to decelerate it early enough to prevent it from falling over.

To implement the tabular Q-learning, we have chosen a two-dimensional state space $\mathbf{x} = (\theta, \omega)$. $30 \times 30 \times 9$ bases were used for the state-action space (θ, ω, u). Each trial was started from an initial state $\mathbf{x}(0) = (\pi, 0.1)$ and lasted 20 seconds. The sample time is 0.03 seconds. As a measure of the swing-up performance, we have chosen t_{up} as the time in which the pendulum stayed up ($|\theta| < \pi/4$). A trial was regarded as "successful" when t_{up} is superior to the t_{up} average of the 1000 last trials.

We tested the performance of the Q-Learning algorithm depending on the shape of the reward function and initial Q-values (Fig. 7). In all cases, the t_{up} average after 10000 trials is around 14 seconds.

We tested first the binary reward function

$$R(\mathbf{x},u,\mathbf{x}') = \begin{cases} 1 \text{ if } |\theta'| < \pi/4 \\ 0 \text{ otherwise} \end{cases} \tag{9}$$

with different uniform initial Q-values in order to observe various behaviors at the beginning of the learning. With $Q_i = 0$, the task was really difficult to learn (`bar1`) because the behavior far from the goal is random. A better performance concerning the binary reward and uniform Q_i was observed with $Q_i > 0$ (`bar2`), *i.e.* the followed behavior when $|\theta| > \pi/4$ is *systematic exploration.* The policy drives the controller to unexplored areas which are assigned higher Q-values, *i.e.* the controller is spurred on to swing the pendulum upwards. Systematic exploration is the best strategy in this specific case.

TABLE I
BETTER CHOICES OF REWARD FUNCTION AND INITIAL Q-VALUES FOR GOAL-DIRECTED RL ROBOT TASKS.

BINARY REWARD FUNCTION FOR DISCRETE STATE SPACE	CONTINUOUS REWARD FUNCTION FOR CONTINUOUS STATE SPACE
$r(s,a,s') = \begin{cases} r_g \text{ if } s' = s_g \\ r_\infty \text{ else} \end{cases}$ Choice of r_g and r_∞ : $r_g \geq \frac{r_\infty}{1-\gamma}$	$r(s,a,s') = \beta e^{-\frac{d(s',s_g)^2}{2\sigma^2}}$
Choice of uniform initial Q-values : $Q_i = \frac{r_\infty}{1-\gamma} + \delta$	Choice of uniform initial Q-values : $Q_i = \frac{\beta}{1-\gamma}$
Choice of goal biased initial Q-values : $Q_i(s) = \beta e^{-\frac{d(s,s_g)^2}{2\sigma^2}} + \delta + \frac{r_\infty}{1-\gamma}$	Choice of goal biased initial Q-values : $Q_i(s) = \beta(1 + \frac{1}{1-\gamma}) e^{-\frac{d(s,s_g)^2}{2\sigma^2}} + \delta$
$\delta \geq 0$ fixes the level of systematic exploration far away from the goal and $\beta > 0$ adjusts the amplitude of the gradient σ specifies the gradient influence area and γ is the discount factor, s is the previous state, s' the new state, s_g the goal state	

Then, we tested goal bias with binary rewards, so $Q_i(\mathbf{x}) = \beta e^{-\frac{\theta^2}{2\sigma^2}} + \delta$. Given our previous results, it is obvious that the goal bias function shall favor systematic exploration when $|\theta| > \pi/4$, so we chose $\delta = 0.1$, $\beta = 1$ and $\sigma = 0.25$ (`bar3`). Goal bias does not improve obviously the learning.

The system is a continuous space control task so the classical reward [13] is given by the height of the tip of the pendulum, *i.e.*, $R(\mathbf{x}, u, \mathbf{x}') = \cos(\theta')$ and the arbitrary Q_i value is 0 (`bar4`). Thus, the pendulum moves round in circles when $|\theta| < \pi/2$ and explores systematically when $|\theta| > \pi/2$ at the beginning. The result is disappointing.

We applied our continuous reward function (7) with $Q_i = 10$ and $\beta = 1$. The distance is defined as $d(\mathbf{x'}, \mathbf{x}_{up}) = \theta'$ with $\mathbf{x'}$ the new state and $\mathbf{x}_{up}$ the goal state. So the continuous reward function is :

$$R(\mathbf{x}, u, \mathbf{x}') = e^{-\frac{\theta'^2}{2\sigma^2}} \tag{10}$$

$\sigma = 0.25$ so that the reward gradient influence area is only around $|\theta| < \pi/4$. Results (`bar5`) are near the case of a binary reward and systematic exploration (`bar2`).

Lastly, we tried goal bias with continuous reward function. The reward function (10) is the continuous equivalent to the binary one so we have kept this one. The reward function is continuous, it is the same for Q_i which must be higher than $\frac{R(\mathbf{x})}{1-\gamma}$. So goal bias is $Q_i(\mathbf{x}) = \beta(1 + \frac{1}{1-\gamma}) e^{-\frac{d^2}{2\sigma^2}} + \delta$. We have kept previous choices: $\delta = 0.1$, $\beta = 1$ and $\sigma = 0.25$. This last simulation (`bar6`) is the better performance concerning the pendulum, with a learning speed under 1000 trials.

VI. CONCLUSION

In this paper, we have presented some rules for speeding up RL within the context of *goal-directed robot tasks*. The main feature of our analysis is that the choice of the reward function and initial Q-values can have a tremendous impact on the performance of RL algorithms. Notably, some values of Q_i lead to a detrimental behavior that must be avoided. Thanks to our experiments, we have confirmed the presence of bounds which mark out diverse behaviors. It is worth noticing that the farther Q_i is from the bounds, the more the characteristic behaviors are distinguished.

Moreover, we advise to be wary of potential-based shaping or progress estimators that may entail pernicious behavior. A *safer adjustable continuous reward function* is also suggested. At last, thanks to our conditions on the initial Q-values, we developed a *generic goal bias function*, whose main feature is to be transient. Table I recapitulates the better choices of reward function and initial Q-values for goal-directed RL. We believe that our method shows the promise of *implementing more efficiently RL algorithms on real goal-directed robot tasks*.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, 1998.

[2] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge University, Cambridge, England, 1989.

[3] S. Iida, M. Kanoh, S. Kato, and H. Itoh, "Reinforcement learning for motion control of humanoid robots," in *Proc. of IROS*, Sendai, 2004, pp. 3153–3157.

[4] S. Waslander, G. Hoffmann, J. Jang, and C. Tomlin, "Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning," in *Proc. of IROS*, Edmonton, 2005.

[5] G. Laurent and E. Piat, "Learning mixed behaviours with parallel q-learning," in *Proc. of IROS*, Lausanne, 2002.

[6] W. Smart and L. Kaelbling, "Effective reinforcement learning for mobile robots," in *Proc. of ICRA*, 2002, pp. 3404–3410.

[7] S. Behnke and M. Bennewitz, "Learning to play soccer using imitative reinforcement," in *Proc. of the ICRA Workshop on Social Aspects of Robot Programming through Demonstration*, Barcelona, April 2005.

[8] M. J. Mataric, "Reward functions for accelerated learning," in *Proc. of the 11th ICML*, 1994, pp. 181–189.

[9] J. Randlov and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proc. of the 16th ICML*, 1998, pp. 463–471.

[10] E. Wiewiora, "Potential-based shaping and Q-value initialization are equivalent," *Journal of Artificial Intelligence Research*, vol. 19, pp. 205–208, 2003.

[11] G. Hailu and G. Sommer, "On amount and quality of bias in reinforcement learning," in *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, Tokyo, Oct. 1999, pp. 1491–1495.

[12] C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.
[13] K. Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, vol. 12, no. 1, pp. 219–245, 2000.